Applications of Deep Learning in GIS - Spatiotemporal data mining and forecasting

Thesis Presentation

Student: Supasin Wuthikulphakdi, TKU#608785068

Advisor: Yihjia Tsai Ph.D.

Contribution

- In ECE), I compared as many models as I could.
  - DEEPFORECAST multi-LSTM is the best
    - 1.575 RMSE, 1.159 MAE
  - Among competing models, more complex ones aren't always better.
    - Proven by metrics to be shown
  - PEMS-BAY trained STDL models better than METR-LA.
- In my CEHE,
  - I chose familiar models of each domain (S/T) to train with my fire event dataset
  - <u>VAE is the best</u> in the space domain
    - 0.026348 RMSE, 0.009244 MAE
  - <u>FBProphet is the best</u> in the time domain
    - 6.97231 RMSE, 5.045342 MAE
  - LSTM doesn't always beat FBProphet

Motivation

Intelligent transport system (ITS)

Disaster & Weather forecast

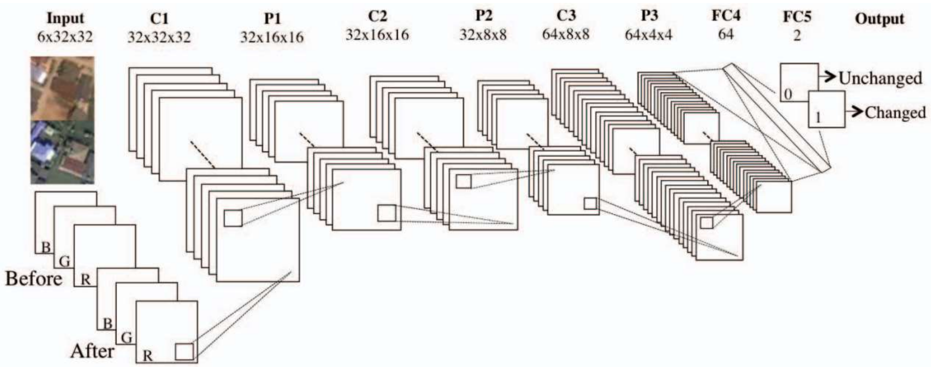
To study DL-Implementations in STDM

To mine ST-data is laborious

- Deep learning can help -> self-feature extraction

Do newer STDM-DL models outperform older ones?

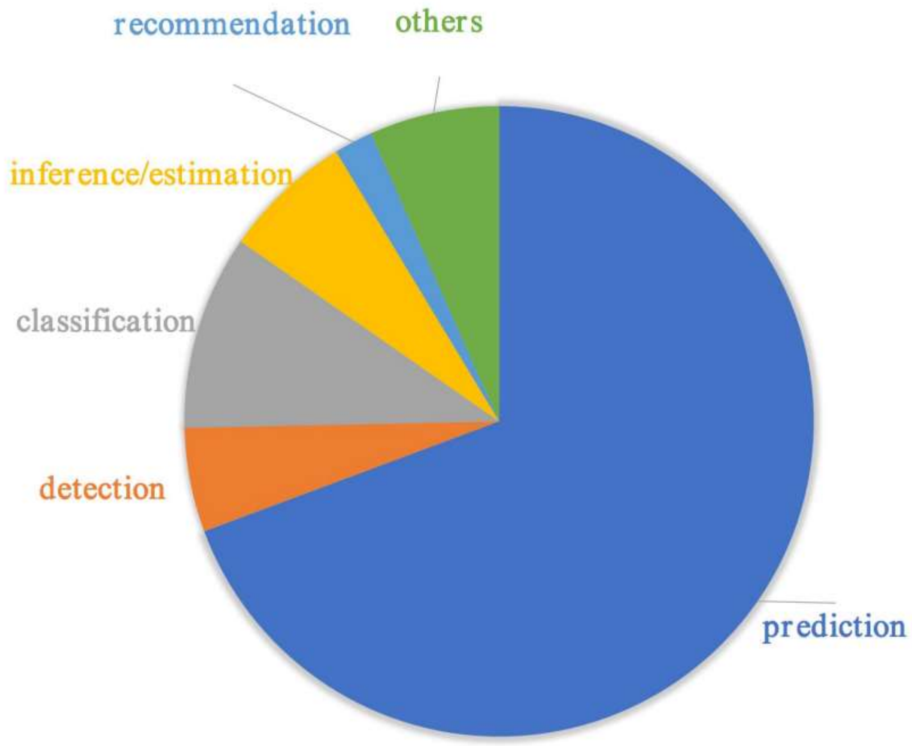
Uses of STDM



- Meteorology
- Transportation (Most popular)
  - ITS
- On-Demand Services
  - Taxi-hailing apps
- LBSN
- Criminology
- Etc.

Literature Review - Early WORKS

- Disaster Impact Analysis & Detection
  - Doshi et al.'s CNN [1,2]
  - Amit & Aoki FC-CNN [3]
  - Fang's LSS-LSTM [6]
- GEO-Feature Detection
  - Iglovikov et al.'s UNET [4]
- Earlier surveys
  - Wang et al. [8]
  - Atluri et al. [19]



Literature Review – Implemented models

2.2 (ECE)

- DCRNN [7]
- STGCN [9]
- ST-METANET[16]
- CRANN [15]
- Adaptive GCNN [71]
- Attention-based ST-GCN [72]
- Deepforecast Multi-LSTM[73]
- Spacetimeformer [80]

2.3 (CEHE)

- Autoencoders
  - BAE
  - VAE
  - CVAE
- GAN
  - DCGAN
  - WGAN
  - LSGAN
- FBProphet
- ARIMA
- LSTM
  - MMS = Matrix Manipulation System

Hypothesis

- Q1: STDM DL-models can do a variety of learning and predictions, but how well can they do?
  - Metrics: MSE, RMSE, & MAE
- Q2: If I have a custom dataset with its data structure visualised, which model to be learned from it?
  - Such as the “NTPC-fire_2015-17” dataset
    - Which is small.
  - What if I vary some hyperparameters?

Experiment Episode 1: Exploratory Comparative Experiment (ECE)

- Hyperparameters: default
- Optimiser: Adam
- Environment: COLAB GPU, TWCC
- Method: run models, record and compare their metrics.

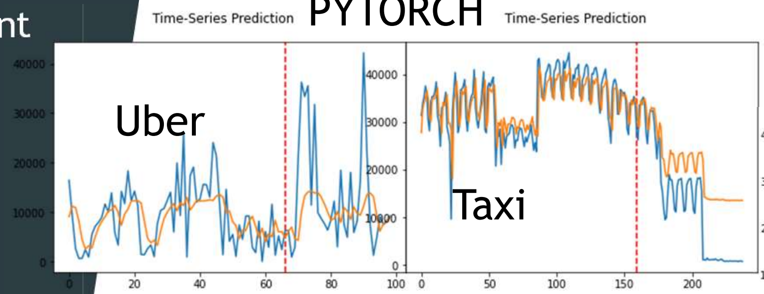
Architecture	Architecture Description	Dataset Name/Desc	learning rate	Epochs	Major DL Module
Taxi-Simple-LSTM-Pytorch	Simple-LSTM	Time series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Uber-Simple-LSTM-Pytorch	Simple-LSTM	Time series of Taxi-Uber DS. (2014-15)	0.01	2000	Pytorch
Taxi-Simple-LSTM-Keras	Simple-LSTM	Time series of Taxi-Uber DS. (2014-16)	0.001	100	TensorFlow Keras
Uber-Simple-LSTM-Keras	Simple-LSTM	Time series of Taxi-Uber DS. (2014-17)	0.001	100	TensorFlow Keras
CRANN-Temporal	Bahdanau Att.Mech Autoencoder (LSTM based)	temporal time series of hourly/daily car traffic (in Madrid)	0.01	200	Pytorch
CRANN-Spatial	CNN+ST-Att.Mech	graph data captured by 30 sensors + Timestamps (A 17000x30 matrix)	0.01	200	Pytorch
CRANN-Dense	Fully Connected Feedforward NN (FCFFNN)	dense 3D+ tensor of both preceding modules	0.01	200	Pytorch
Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
GAT Seq2seq (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
ST-Metanet (flow)	Improved Seq2eq	Beijing TDrive	0.01	200	MXNET
Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
GAT Seq2seq (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
ST-Metanet (speed)	Improved Seq2eq	METR-LA	0.01	200	MXNET
AGCRN	Attentive Graph CRN	Caltrans PEMS04&08	0.003	100	Pytorch
ASTGCN	Attention Based GCN	Caltrans PEMS04&08	0.001	80	Pytorch
Deepforecast	Multi-LSTM	MS_winds - Wind Speed & Flow Dataset	0.001	80	TensorFlow Keras
DCRNN	R-CNN	PEMS & METR-LA	0.001	100	TensorFlow Keras
STGCN	Graph-CNN	PEMS & METR-LA	0.001	50	Pytorch
Spacetimeformer	Transformer opted for ST-data	METR-LA	0.001	80	Pytorch

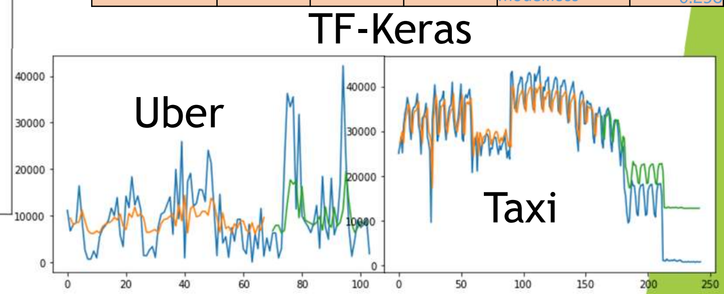
ECE results:

- Deepforecast Multi-LSTM
  - Is the best model to date
  - 1.575 RMSE, 1.159 MAE
- All model performed well
  - At least for their arch.
- LSTM on the TAXI-Uber data
  - Had a poor metrics
  - Simplicity, overfitting, gradient vanishing & explosion

		Key Metrics		Other Metrics	
Architecture	MSE	RMSE	MAE	Type	Value
Taxi-Simple-LSTM-pytorch	72269860	8501.168	5753.569		
Uber-Simple-LSTM-pytorch	0.0332864	0.1824457	0.13018544		
Taxi-Simple-LSTM-Keras	93431556	9666	92.484945		
Uber-Simple-LSTM-Keras	105050200.4	10249.4	83.21		
CRANN-Temporal	6653.6636	81.569992	50.3727684	Bias	-3.82883501
				Relative error %	24.85518008
CRANN-Spatial	55740.719	236.09472	117.7406616	Bias	-14.2317371
				Relative error %	9.053748846
CRANN-Dense	67264.055	259.35315	138.2879333	Bias	-4.71734381
				Relative error %	24.85518008
Seq2seq (flow) [14]	1626.986623	40.33592224	21.3		
GAT Seq2seq (flow) [14]	1098.041371	33.13670731	18.3		
ST-Metanet (flow) [14]	813.1885148	28.51646042	16.9		
Seq2seq (speed) [14]	44.4751941	6.668972492	3.55		
GAT Seq2seq (speed) [14]	36.92343427	6.076465607	3.28		
ST-Metanet (speed) [14]	33.63158961	5.799274921	3.05		

		Key Metrics		Other Metrics	
Architecture	MSE	RMSE	MAE	Type	Value
AGCRN - PeMSD4	1040.761367	32.26083333	19.69416667	MAPE (TWCC)	13.02040833
AGCRN - PeMSD8	718.9101563	26.8125	17.01583333	MAPE(COLAB)	10.65138333
ASTGCN - PeMSD4	1173.7476	34.26	21.84	MAPE	0.15
ASTGCN - PeMSD8	809.4025	28.45	18.5	MAPE	0.11
Deepforecast	2.481881285	1.5753988	1.1590073	NRMSE_maxmin (%)	15.575216
				NRMSE_mean(%)	43.097104
DCRNN(Metr-LA)-STA - 15min	75.5161	8.69	4.02	MAPE	9.39
DCRNN(Metr-LA)-STA - 1hr	201.9241	14.21	6.79	MAPE	16.71
DCRNN(Metr-LA)-VAR- 15min	60.5284	7.78	4.37	MAPE	10.08
DCRNN(Metr-LA)-VAR - 1hr	114.0624	10.68	6.5	MAPE	15.84
DCRNN(Pemsbay)-STA - 15min	11.7649	3.43	1.6	MAPE	3.24
DCRNN(Pemsbay)-STA - 1hrmin	49.2804	7.02	3.05	MAPE	6.84
DCRNN(Pemsbay)-VAR - 15min	9.5481	3.09	1.74	MAPE	3.59
DCRNN(Pemsbay)-VAR - 1hr	26.1121	5.11	2.92	MAPE	6.46
STGCN - 15min	16.532356	4.066	2.231	%Wmape	5.206
STGCN - 30 min	33.051001	5.749	3.04	%Wmape	7.386
STGCN - 45min	46.744569	6.837	3.623	%Wmape	8.927
Spacetimeformer [80]	36.21	6.017474553	2.82	MAPE	7.71
				model.loss	0.258





CEHE – Procedure

Download & visualise the dataset

Extract spatial (coordinates) and temporal features.

- Spat.F. = coordinates, temp.F. = timestamps with one frequency each

Convert coordinates to rasterised heatmaps

- And temporal features to time series

Run spatial models on rasterised heatmaps

- And temporal models on time series

Visualise and record the results

EP2: Custom Event Heatmap Experiment (CEHE)
Hyperparameters (HPs):
- Epochs: 100
- Optimiser: Adam, LR = 0.001
- OTHERWISE, default according to their origins.

- *In the spatial domain*
- *VAE is the Best*
  - *Due to moderate simplicity* & well tuned HPs despite minimal touches
- Due to my small data size (36samples, 36*784 rpsns)
  - GANs did not beat the AEs
- *In the temporal domain (1096 samples)*
- *FBProphet is the best*
  - *Never overfits but can be overtaken by PreMMS LSTM if well tuned.*
- *Hidden subtest:*
  - *Add lookback steps (+)*
  - *Add more LSTM cells (+)*

MODEL	DOMAIN	Batch size	Train Metrics:				Test Metrics:			
			MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
PreMMS LSTM4			91.630625	9.572389	5.251044	0.244517	50.876132	7.132751	5.399653	0.369033
PostMMS-LSTM4-noLB	Time	1LB	7.679134	2.771125	1.556069	0.133908	61.438013	7.83824	5.954068	0.29369
PostMMS-LSTM-LB5	Time	5LB	7.582465	2.753628	1.559106	0.138684	57.692435	7.595554	5.70559	0.283949
BAE	Space	24tr + 12te	0.00280751	0.05298596	0.04602381	38999060	0.00357825	0.05981846	0.053716972	46504212
DCGAN	Space	9	0.02931224	0.17120818	0.04847476	1127299.4	0.02781864	0.16678922	0.0468716	598425
FBProphet (37 days)	Time	1 day	n/a	n/a	n/a	n/a	48.613108	6.97231	5.045342	0.281977
CVAE	Space	9	0.00515399	0.07179128	0.01000213	4424340	0.00433511	0.06584153	0.009183249	4546362.5
VAE	Space	9	0.00083985	0.02898009	0.009215273	2282401.2	0.00069422	0.02634812	0.00924397	1.30263E+13
WGAN-GP	Space	9	0.00812537	0.09014085	0.021708265	8.3067E+11	0.00754667	0.08687156	0.02071965	7.40703E+11
Weekly ARIMA	Time	7 days	n/a	n/a	n/a	n/a	1436.90057	37.9064714	28.42712296	0.195063926
LSGAN	Space	9	0.1126827	0.33568244	0.18267849	0.57419306	0.10933066	0.33065187	0.18496665	0.5471558
PostMMS-LSTM-Weekly	Time	7 days	227.370864	15.078822	10.250484	0.086445	852.383545	29.195608	21.007481	0.143819
PostLSTM9	Time	5LB	7.578619	2.752929	1.563434	0.139068	55.893509	5.60772	5.70559	0.281344

PreMMS-LSTM

```
#Form dataset matrix
def create_dataset(df, previous=1):
    dataX, dataY = [], []
    for i in range(len(df)-previous-1):
        a = df[i:(i+previous), 0]
        dataX.append(a)
        dataY.append(df[i + previous, 0])
    return np.array(dataX), np.array(dataY)

#tensor manipulation
tseries = dftempday['freq'].to_numpy()
dftensor=pd.DataFrame(tseries) #forward dataframe
dftensor=np.array(dftensor).astype('float32') #convert to tensor

#[PRE-MMS] normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dftensor)

# split into train and test sets
#here, you can change the test size to make future prediction
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]

# reshape into X=t and Y=t+1
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
lookback = look_back

# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

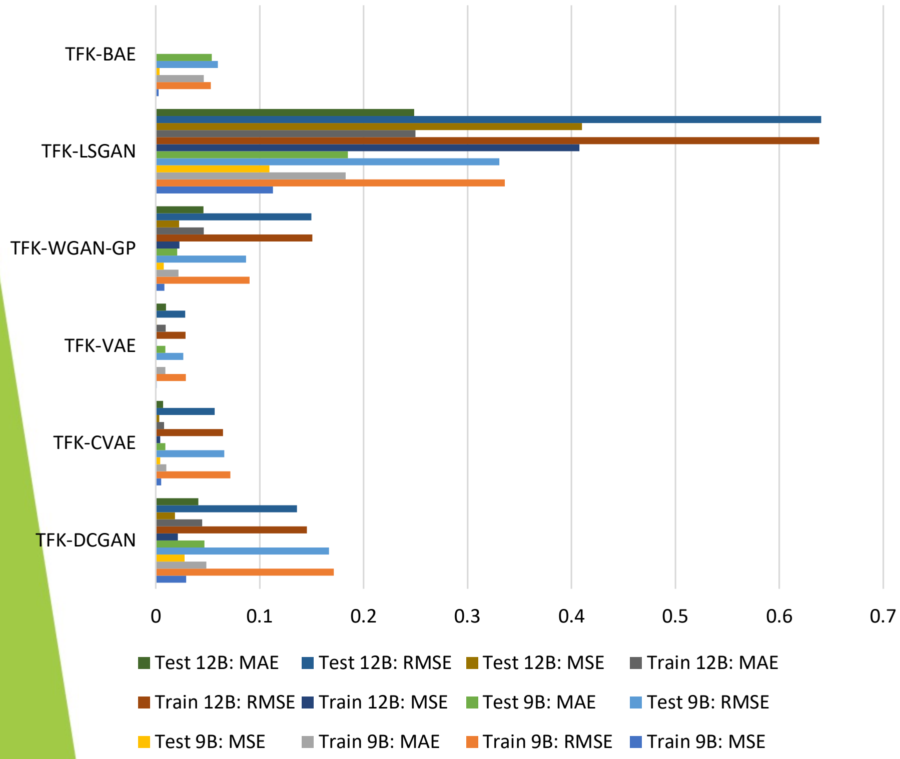
PostMMS-LSTM

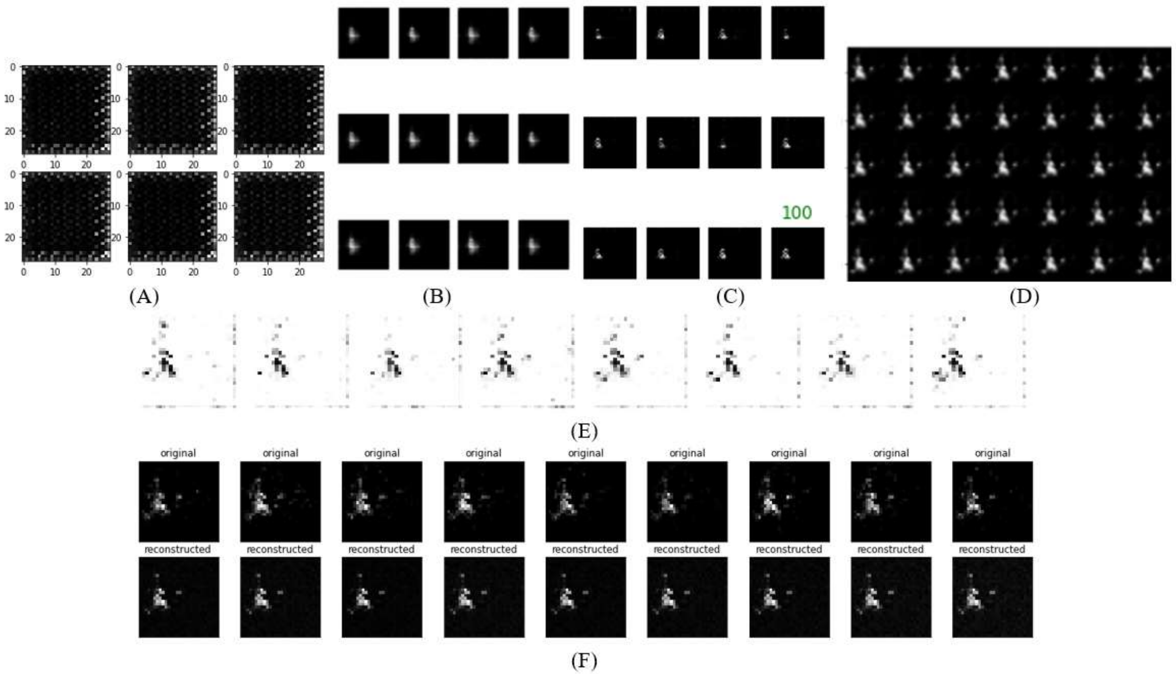
```
#Form dataset matrix
def create_dataset(df, previous=1):
    dataX, dataY = [], []
    for i in range(len(df)-previous-1):
        a = df[i:(i+previous), 0]
        dataX.append(a)
        dataY.append(df[i + previous, 0])
    return np.array(dataX), np.array(dataY)

#tensor manipulation
tseries = dftempday['freq'].to_numpy()
dftensor=pd.DataFrame(tseries) #forward dataframe
dftensor=np.array(dftensor).astype('float32') #convert to tensor

#train-test split
train_size = int(len(dftensor) * 0.8)
test_size = len(dftensor) - train_size
train, test = dftensor[0:train_size,:], dftensor[train_size:len(dftensor),:]

#[POST-MMS] minmaxscaler
scaler = MinMaxScaler(feature_range=(0, 1))
train = scaler.fit_transform(train)
test = scaler.fit_transform(test)

# Lookback period
lookback = 5
trainX, trainY = create_dataset(train, lookback)
testX, testY = create_dataset(test, lookback)

# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Spatial domain analysis :The Batch Size test

- Expand the training batch (9 -> 12)
  - DCGAN & CVAE improved
  - WGAN-GP & LSGAN worsened
  - VAE
    - Somewhat improved train MSE
    - While slightly worsened others
    - Is the best spatial model

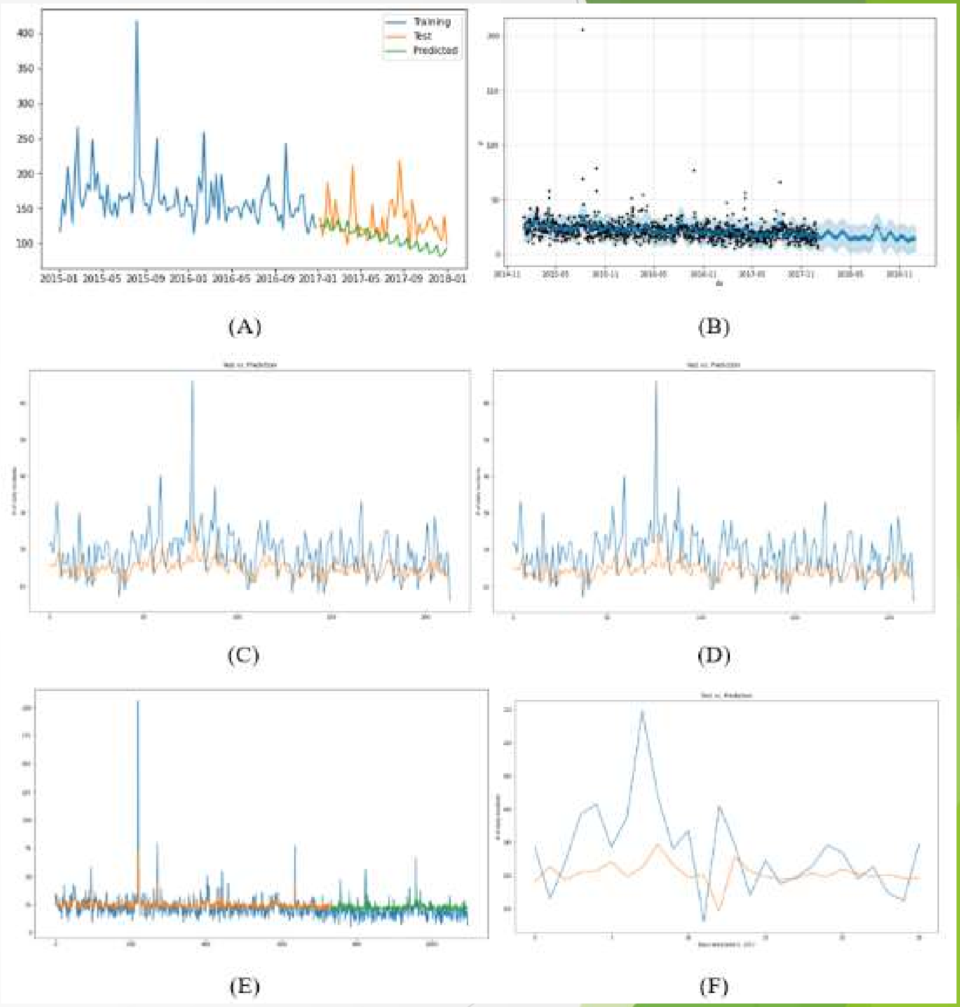
		TFK-DCGAN	TFK-CVAE	TFK-VAE	TFK-WGAN-GP	TFK-LSGAN	TFK-BAE
Train 9B:	MSE	0.02931224	0.00515399	0.00083985	0.00812537	0.1126827	0.00280751
	RMSE	0.17120818	0.07179128	0.02898009	0.09014085	0.335682439	0.05298596
	MAE	0.04847476	0.01000213	0.00921527	0.02170827	0.18267849	0.04602381
Test 9B:	MSE	0.02781864	0.00433511	0.00069422	0.00754667	0.10933066	0.00357825
	RMSE	0.16678922	0.06584153	0.02634812	0.08687156	0.330651874	0.05981846
	MAE	0.0468716	0.00918325	0.00924397	0.02071965	0.18496665	0.05371697
Train 12B:	MSE	0.02113337	0.00419831	0.00082212	0.02267304	0.40772453	
	RMSE	0.14537322	0.06479436	0.02867267	0.15057571	0.638533108	
	MAE	0.0444729	0.00805869	0.00947048	0.04603831	0.24995048	
Test 12B:	MSE	0.01843591	0.003209	0.00079672	0.02239976	0.41008073	
	RMSE	0.13577889	0.05664803	0.02822621	0.14966548	0.640375461	
	MAE	0.04102352	0.00706938	0.00987247	0.04573417	0.24851514	





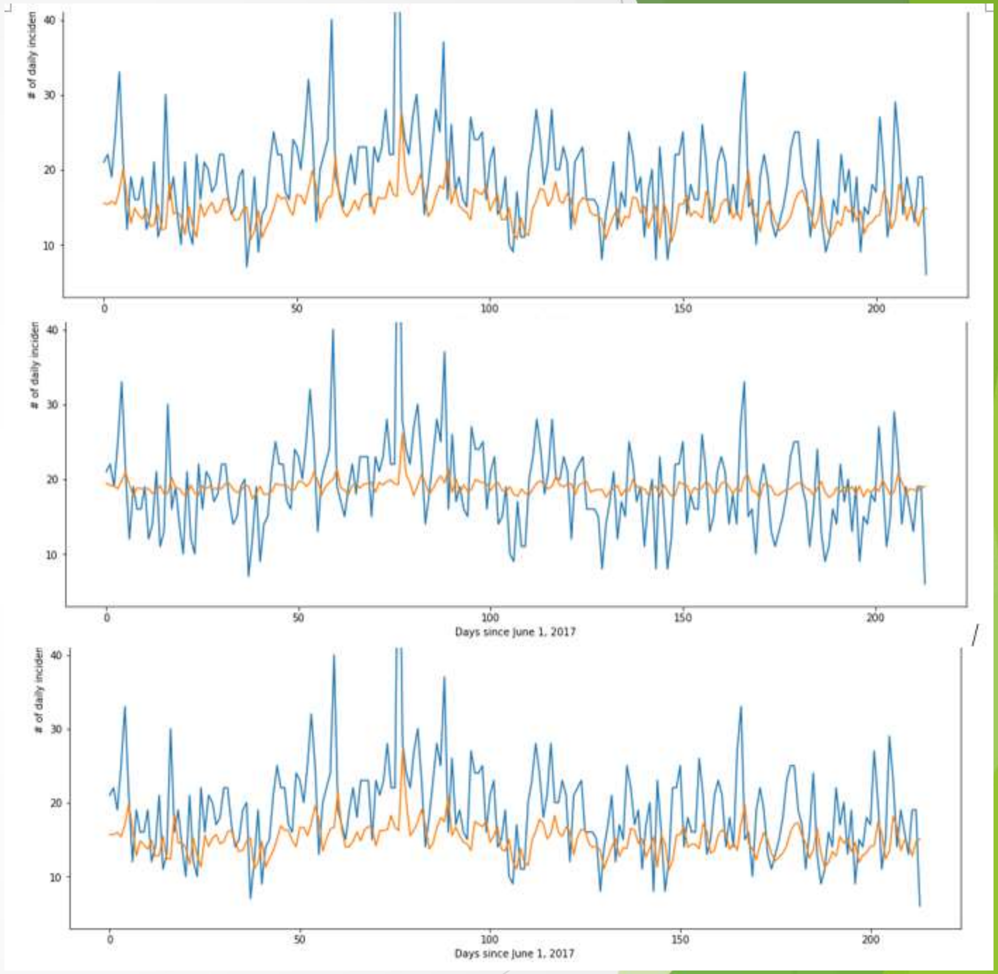
Visualised Frequency Prediction

- A = Weekly Auto ARIMA
- B = FBProphet
- C = MMS-LSTM-5LB
- D = MMS-LSTM-noLB
- E = Vanilla LSTM
- F = LSTM-Weekly



What if I use the weekly model to test the daily data

- Top = LSTM4
- Middle = LSTM4 trained with weekly series
- Bottom = LSTM9



Findings of my Thesis & Discussion (1/2)

- The ECE taught me that
  - ST-METANET, Pems-DRCNN, STGCN & Spacetimeformer performed similarly well.
  - PEMS bay train each model better than METR-LA
  - Each model did well for its architecture.
- The Spat. CEHE taught me that
  - AEs are better that GANs for smaller datasets
    - GANs are harder to train & require more data than AEs
  - Due to the much larger representation size (S:28224(36*28*28) vs T:1096) and milder gradient
    - Spatial models are much better than temporal ones, as they experienced much larger data
- Future plans & recommendations
  - More LSTM/AE/GAN variants
  - Daily rasterisation alg.
  - Do more subtests
    - Vary more HPs
    - broader HP-ranges
- The Temp. CEHE taught me that
  - As long as FBProphet is better, never trust the LSTM.
  - It's possible that vanilla LSTM will defeat FBProphet (if well tuned)
  - The more frequent a series, the better a model trains.
- The ST-METANET & Spacetimeformer
  - Have a RAM problem
- Finally, I've learned that
  - Newer, more complex models doesn't always outperform older, more simple ones
  - The subtests will eventually lead to
    - "HP-Tuning"

Findings of my Thesis & Discussion (2/2)

- Open questions
  - Is there any model that are good and dealing with CEHE dataset BOTH in spacial and temporal domain?
    - Currently no, because the datasets of most of the explored models involve sensors recording values at each timestamp
    - But my dataset has no sensors, only list of coordinates with only 1 frequency each. And thus, must be rasterized to predict/reconstruct heatmaps.
  - If you combine the best in temporal model and the best in spatial model for CEHE dataset, what would be the model looked like?
    - VAE-FBP, VAE-LSTM

References

- [1] Doshi, J., Basu, S. and Pang, G., 2018. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*.
- [2] Doshi, Jigar. 2018. Residual Inception Skip Network for Binary Segmentation. Pages 216–219 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

  [3] Amit, S.N.K.B. and Aoki, Y., 2017, September. Disaster detection from aerial imagery with convolutional neural network. In 2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC) (pp. 239-245). IEEE.
- [4] V. Iglovikov, S. Mushinskiy, and V. Osin, "Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition," vol. June 2017.
- [6] Fang, Z., Wang, Y., Peng, L. and Hong, H., 2020. Predicting flood susceptibility using LSTM neural networks. *Journal of Hydrology*, [online] p.125734. Available at: <https://doi.org/10.1016/j.jhydrol.2020.125734> [Accessed 18 April 2021].
- [7] Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.
- [8] S. Wang, J. Cao, & P. Yu, "Deep Learning for Spatio-Temporal Data Mining: A Survey," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.3025580.
- [9] Yu, B., Yin, H., & Zhu, Z. (2017). Spatiotemporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.
- [15] de Medrano, R., & Aznarte, J. L. (2020). A spatiotemporal attention-based spot-forecasting framework for urban traffic prediction. *Applied Soft Computing*, *96*, 106615.
- [16] Luo, X., Li, D., & Zhang, S. (2019). Traffic flow prediction during the holidays based on DFT and SVR. *Journal of Sensors*, *2019*.
- [19] Atluri, G., Karpatne, A., & Kumar, V. (2018). Spatiotemporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)*, *51*(4), 1-41.
- [71] Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *arXiv preprint arXiv:2007.02842*.
- [72] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019, July). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 922-929).
- [73] Ghaderi, A., Sanandaji, B. M., & Ghaderi, F. (2017). Deep forecast: Deep learning-based spatiotemporal forecasting. *arXiv preprint arXiv:1707.08110*.
- [80] Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-Range Transformers for Dynamic Spatiotemporal Forecasting. arXiv preprint arXiv:2109.12218.

Discussion & Conclusion: What have I done?

- Technical basics
  - Applications, DL-Techniques
- Paper survey
  - SOTA models are compared in my ECE.
  - Common models are implemented to my fire event data
- Comparison tests
  - ECE
  - CEHE
    - How STDM works?
    - Rasterised Heatmap Generation
    - Time series prediction
    - HP subtests
- Discovered facts
  - (Newer, more complex vs older, more simple models)
  - [BEST ECE] DEEPFORECAST multi-LSTM
  - [BEST CEHE] VAE & FBProphet
- Discussion
  - Gave future subtest recomms.
  - Listed some open problems
  - Further discuss the results
  - Pointed out current issues
  - Indicated future research directions.

Question Time!

Q&As.